A Review on Conformance Checking Technique for the Evaluation of Process Mining Algorithms

Vahideh Naderifar, Shahnorbanun Sahran, Zarina Shukur

Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

Abstract **– Information systems ease the collecting, processing, storing, and distributing informations, which in turn lead to decision-making and control in organizations. Information systems consist of information about key people, locations, and other important things in the organization. There is insufficient research on process mining, and the connection between data mining and model-driven process management. The goal of process mining is to discover, monitor, and improve processes by extracting knowledge from event logs. Conformance checking compares a process model with its event log. The inputs are event log and process model, while the output consists of diagnostic information, which shows differences and commonalities between the model and its log. The conformance checking technique is able to identify the similarities or differences between the process model and the event log. This research will introduce the conformance checking measures (fitness, precision, simplicity, and generalization) in order to improve the process model discovering from analytical aspects.**

Keywords **– process mining, conformance checking, control flow, process model, event log.**

DOI: 10.18421/TEM84-18
https://dx.doi.org/10.18421/TEM84-18

Corresponding author: Zarina Shukur,
Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
Email: *zarinashukur@ukm.edu.my*

Received: 12 April 2019.
Revised: 27 August 2019.
Accepted: 03 September 2019.
Published: 30 November 2019.

1. Introduction

Process mining is a process management technique that allows the analysis of business processes based on event logs. An event log can be produced from information systems. Information systems ease the collecting, processing, storing, and distributing information, which leads to decision-making and control in organizations. Information systems consist of information regarding key people, locations, and other important things in the organization. Typically, these systems try to discover more information from events and record them during the execution process. Process mining is a consideration on the observation of the process during execution to the analytical model of the process. The goal of process mining is to discover, monitor, and improve processes by extracting knowledge from their event logs. Process mining automatically discovers process models without having a priori model based on event logs. The process mining method can be used to solve problems faced by organizations, for instance, the problem of having low information on workflow in the organization. Figure 1 illustrates the concept of process mining.

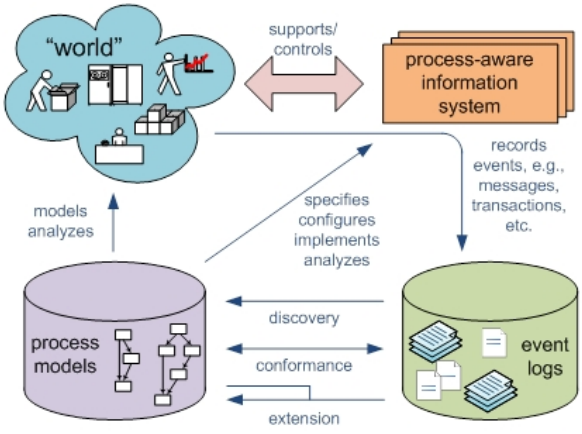


Figure 1. Concept of process mining
Source: Van der Aalst et al. (2012)

There are three classes of process mining technique. This classification is based on whether

there is a prior model and, if so, how it is used; **discovery**, **conformance checking**, and **enhancement**. The **discovery technique** takes an event log and constructs a model.

- **Discovery**

The first class of process mining is discovery. Process discovery generates a process model from the event log without a priori model. The discovered model is presented with process model languages, such as Petri net or BPMN. A majority of researchers focus on process discovery, which bring about many process discovery techniques in this field.

- **Conformance checking**

Conformance checking compares a process model with its event log. The inputs are event log and process model, while the output consists of diagnostic information showing differences and commonalities between the model and the log. The conformance checking technique is able to identify the similarities and differences between the process model and the event log, and whether there is a good match between the process model and event log.

- **Enhancement**

The last class of process mining is process enhancement or process extension. This class assumes the existing process model, and tries to extend or improve the model by adding more data or using the observed events.

Process mining has three types of perspectives: **process perspective**, **organizational perspective**, and **case perspective.** The process perspective focuses on control flow. The control flow characterizes all possible paths among tasks and shows the given behaviours in the log via graphical model. The diagram can help to determine the information about activities, the relationship between them, which tasks are running, and whose activity is related to the certain case. The control-flow mining techniques must be constructed from correct mining. From the aspect of common control flow, correct mining means that constructs can appear as a process model with each language notation. These constructs are sequences, concurrency, loops, non-free-choice, invisible tasks, and duplicate tasks. We can note that activity or task has the same meaning throughout this study.

As mentioned previously, process models present a relation between activities with graphical languages [29],[30]. There are various language formalisms for the definition of processes. These languages mutually describe processes in terms of activities. There are many process modelling languages in process mining.

One of them is Petri Net, which was introduced by [18].

A Petri net is a tuple (P, T, W) where P and T represent finite sets of places and transitions, respectively, with $P \cap T = \emptyset$. In addition, the relation $W \subseteq (P \times T) \cup (T \times P)$ is a finite set of directed arcs.

2. Related Works

Historically [8],[3],[17] were the ones who worked on the process discovery challenges. [8] were among the first researchers on process mining discovery in 1995. They mined a model from the event log in the area of software engineering and called it “process discovery”. [3] were among the researchers from the business fields who introduced a discovery mined model. They called their model “work flow mining”. [17] extended and improved [3]’s algorithm. Since then, many authors have focused on process mining, and provided new algorithms that are able to obtain better results. [12], [13] addressed the issue of process mining in the approach described in their studies, which also allowed concurrency. [20] provided an α-algorithm for business process mining by analyzing the ordering relations between activities in the event log. [23], [22] developed a robust, heuristic-based method for process discovery known as heuristics miner. Heuristics miner can discover short loops, but it is not capable of detecting non-local, non-free–choice constructs. Moreover, the heuristics miner cannot detect duplicate activities. [4] extended the α-algorithm to mine short loops, known as α+.

[5],[6],[7] provided a genetic-based algorithm that is able to deal with the following structural patterns: sequence, choice, parallelism, loops, non-free-choice, invisible tasks, and duplicate tasks. [24],[25] presented two extensions for the α-algorithm .The first extension was β-algorithm, [24], that could tackle concurrency and short loops. The α++ algorithm was a second extension of the previous algorithm by [25], [24]. It described the process model by Petri nets with non-free-choice and non-local structures. [27] presented a process discovery algorithm by using integer linear programming (ILP) that was able to handle loops of activities and non-free–choice structures. An LIP miner creates a flower model. [14] presented a novel technique for data-aware process mining that focused on data flow by using a decision miner approach provided by [16]. Their method could address an invisible transition and a loop to discover the XOR-splits/joins. [26] presented a process mining approach based on the heuristic process discovery technique. The advantages of this algorithm were it could tackle loops and duplicate tasks and could be fitted with the event log. [28] provided a new technique to face the duplicate tasks problem in the process model.

3. Conformance Checking

Conformance checking compares events in the event log with activities in the process model. The goal is to find problems between the modelled behaviour and the observed behaviour. Conformance checking can be used for repair models that are not matched well with their event log. Figure 2 illustrates the main idea of conformance checking.

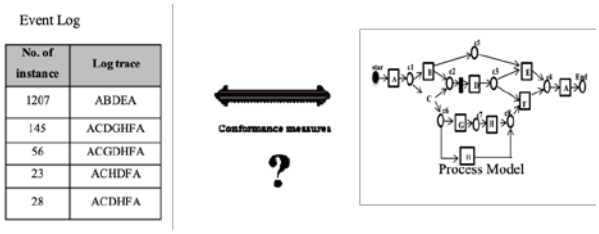


Figure 2. Conformance checking: comparing observed behaviour with modelled behaviour

[15] proposed an incremental approach for assessing two qualities of conformance checking. The first quality was fitness between the process model and log, whereas the second quality was appropriateness of the model and related event logs. Appropriateness can be assessed on both structure (simplicity) and behaviour (precision). [21] presented a quantification approach based on behavioural profiles. Existing approaches for quantifying precision are time-consuming and have problems dealing with non-fitting traces. [1], [2] introduced a power technique for fitness between an event log and a process model of Petri net. Later in 2012, they proposed an alignment-based precision checking technique. They claimed that the most of discovered techniques were unable to deal with non-fitting traces and were time-consuming regarding the quantifying precision.

Conformance checking techniques need an event log and a model as their input. The output consists of diagnostic information showing differences and commonalities between the model and its log. Conformance checking compares a process model with respect to the event log with four main quality dimensions: **fitness**, **simplicity**, **precision**, and **generalization**. Throughout the remainder of this section, examples of event logs in Table 1 and the respective process model in Figure 3 are used to describe how to measure these four quality dimensions.

Table 1: Example of Event Log, L1

Event log	Frequency
ABDEA	1207 times
ACDGHFA	145 times
ACGDHFA	56 times
ACHDFA	23 times
ACDHFA	28 times

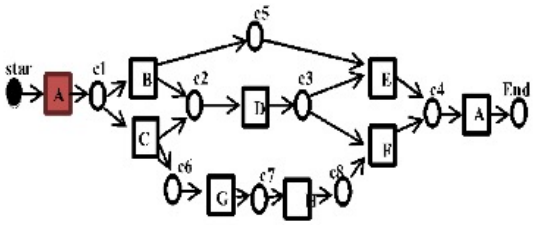


Figure 3. Process model M in terms of Petri net

A. Fitness Calculation

Fitness quantifies, by which the process discovery model can accurately express all behaviours recorded in the event log. The final fitness score is calculated in this study as follows:

$$f(\sigma, N) = \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}\right)$$

Assume L is an event log and N is a WF-net. Note that $\sigma \in L$ is an event sequence of L, $\sum$ denotes the sum of all produced, consumed, missing, and remaining tokens, and applies the same formula. Let pN,σ denote the number of produced tokens when replaying σ on N. mN,σ is the alternative duplicate task that is never repeated together in one sequence number of missing tokens when replaying σ on N. cN,σ is the number of consumed tokens. rN,σ is the number of remaining tokens. If fitness is 1, it means that the discovery process model can replay all traces in the event log.

Example:

The first and second steps are very similar. See Figure 4a.

m-0, r=0, c=0, p=1

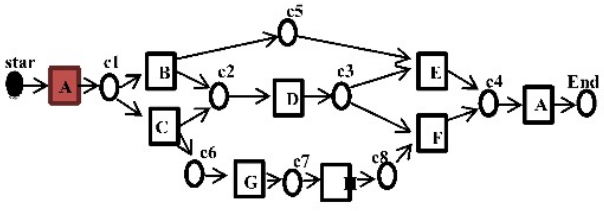


Figure 4a. The first step of the fourth trace of L1 ACHDFA

In the second step, transition "A" is fired, a token from the "Start" place is consumed, and a token at place "C1" is produced. Transitions "B" and "C" are then enabled. According to the fourth trace of the event log, transition "C" should be fired. See Figure 4b.

m=0, r=0, c=1, p=2

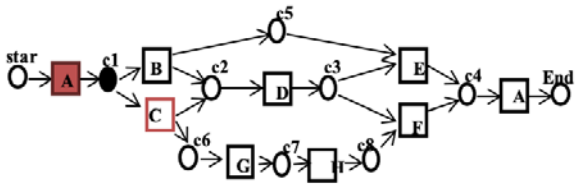


Figure 4b. The second step of the fourth trace of L1
AC*HDFA*

In the third step, transition "C" is fired and then a token from place "C1"is consumed and two tokens are produced into "C2" and "C6". Afterwards, transitions "D" and "G" are enabled to be fired.

m=0, r=0, c = 2, p = 4. See Figure 4c.

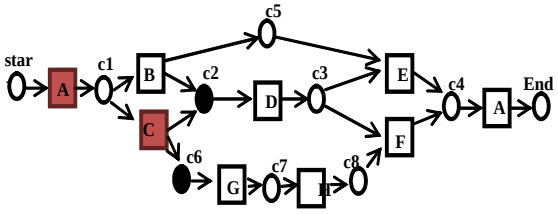


Figure 4c. The third step of the fourth trace of L1
AC*HDFA*

In the fourth step, transitions "D" and "G" are enabled, but according to the trace, transition "H" should be fired. Transition "H" is fired before transition "G", and then a missing token is recorded in place "C6", and a token is created in place "C7" artificially. See Figure 13.

m=1, r=0, c=2, p=4

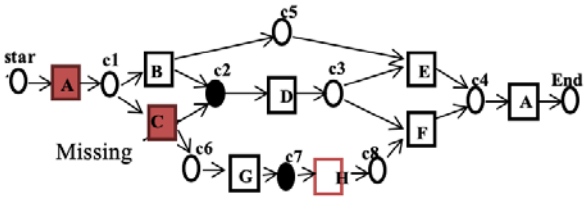


Figure 4d. The forth step of the fourth trace of L1
ACH*DFA*

In the fifth step, transition "H" is fired and a token is consumed from place "C7", while another token is produced in place "C8". See Figure 4e.

m=1, r=0, c=3, p=5

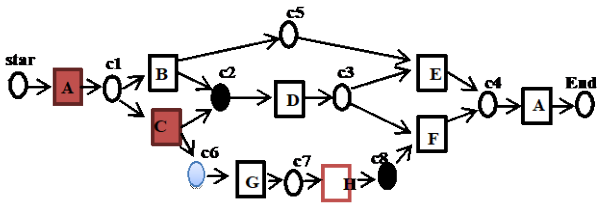


Figure 4e. The fifth step of the fourth trace of L1
ACH*DFA*

In the sixth step, according to the event log, transition "D" should be fired, a token from place "C2" is consumed, and a token at place "C3" is produced. See Figure 4f.

m=1, r=0, c=4, p=6

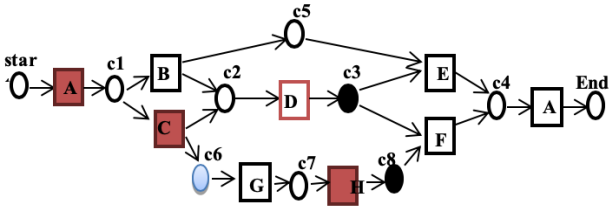


Figure 4f. The sixth step of the fourth trace of L1
ACHD*FA*

In the seventh step, there are two tokens in places "C3" and "C8". Transition F is fired, two tokens are consumed, and a token is produced in place "C4". See Figure 4g.

m=1, r=0, c=6, p=7

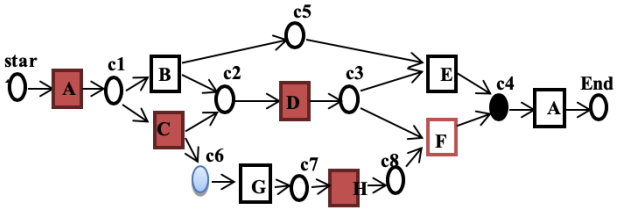


Figure 4g. The seventh step of the fourth trace of L1
ACHDF*A*

In the eighth step, transition A is fired, a token from place "C4" is consumed, and a token is produced in place "End". See Figure 4h.

m=1, r=0, c=7, p=8

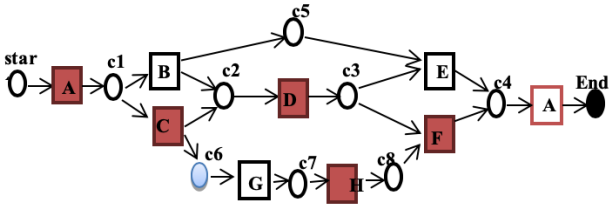


Figure 4h. The eight step of the fourth trace of L1

In the last stage, the token in the "End" place should be consumed. The token is removed from the "End" place. See Figure 4i.

m=1, r=0, c=8, p=8.

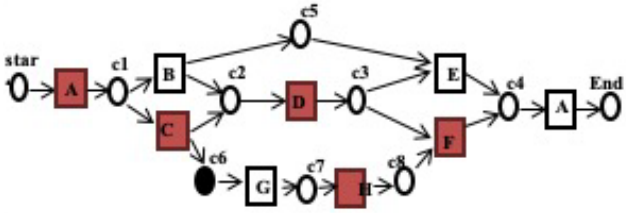


Figure 4i. The last step of the fourth trace of L1

A replay of the fourth trace of event logs is finished as we can see in the figure above. There is still one token in the "C6" place, which means the remaining token is 1. Finally, the tokens will be updated as follows:

m=1, r=1, c=8, p=8

The metric fitness will be as follows

$$f(\sigma 4, L) = \frac{1}{2}\left(1-\frac{1}{8}\right)+\frac{1}{2}\left(1-\frac{1}{8}\right) = 0.87 \quad (2)$$

Fitness =0.87, means that process model M can replay 87% of the observed behaviours of event log L1 correctly. If there are more missing and remaining tokens in the model, then fitness will be poor. The fitness metric is computed for one trace of the event log in the above example. There is a general formula to calculate fitness between the whole event log and process model as follows:

$$f(\sigma, N) = \frac{1}{2}\left(1-\frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}\right)+\frac{1}{2}\left(1-\frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}\right)$$

m_i is the total number of missing tokens for event log and n_i is the frequency of trace i. r_i is the total number of the remaining tokens for event log. p_i is the total number of produced tokens and c_i is the total number of consumed tokens. The total replay fitness as the above example can be calculated as follows:

First trace "σ= ABDEA" numbers of tokens are
m=0, r=0, c=7, p=7
Second trace "σ = ACDGHFA" tokens are
m=0, r=0, c=9, p=9
Third trace "σ = ACGDHFA" tokens are
m=0, r=0, c=9, p=9
Fourth trace "σ = ACHDFA" tokens are
m=1, r=1, c=8, p=8
Last trace "σ = ACDHFA" tokens are
m=1, r=1, c=8, p=8

Fitness can be calculated as:

$$f(M, L) = \frac{1}{2}\left(1-\frac{51}{10666}\right)+\frac{1}{2}\left(1-\frac{51}{10666}\right) = 0.995$$

Event log L1 and process model M have a fitness of 0.995. It means that event log L1 has a 0.005 deviation. Process model M is unable to explain 0.005 of the observed behaviours. Figure 4j shows the process model after the replay of the event log.

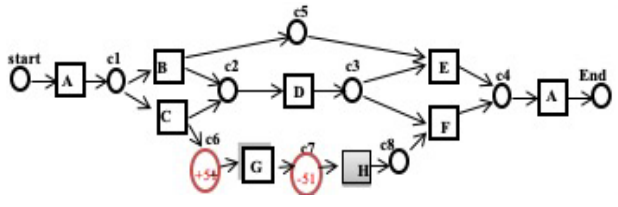


Figure 4j. Diagnostic token illustrates error in the location error

The diagnostic token in Figure 4j illustrates the error in the location error. The "C6" place shows "+51", which means there are 51 tokens remaining (r= 51), and the "C7"place indicates " -51", which means transition "H" has occurred 51 times when it was not enabled. Fitness is one of the dimensions of conformance checking. However, it is unable to solely judge the quality of the process model. A process model with a perfect fitness can be constructed in the way that it can replay all traces in the event log, e.g. flower model (see Figure 5). The flower model has a perfect fitness since it can always replay all observed behaviours of the event log, but it lacks precision. It allows the presence of too many observed behaviours that have never been seen in the event log. Then, fitness is insufficient on its own, and the model will need other quality measures.

B. Precision Calculation

Precision quantifies part of the behaviours are allowed by the model, and they have never been seen in the event log. The model's precision does not allow too many behaviours that do not exist in the logs. The present study used the method by Rozinat and Van der Aalst (2007) based on behavioural appropriateness, which is defined as follows:

$$\alpha_B = 1 - \frac{\sum_{i=1}^{k} n_i x_i}{(m-1)\sum_{i=1}^{k} n_i}$$

Assume t k is the number of different traces from the log. For each log, trace is ($1 \leq i \leq k$), n_i is the number of process instances combined into the current trace, and x_i is the mean number of enabled transitions during log replay, which is calculated from the outgoing edge and used edge. Furthermore, m is the number of labelled tasks (i.e., does not include invisible tasks, and assuming $m>1$)

$$x_i = \frac{outgoing\_edge_i - used\_edge_i}{outgoing\_edge_i}$$

Precision is the second quality in conformance checking. Precision qualifies the preciseness of the process model. A process model that lacks precision shows extra behaviours that do not exist in the event log. If the model is not precise, then it is under-fitting. Under-fitting is a problem whereby the model has too many behaviours that have never been seen in the log. A model that is in a perfect precision does not allow for too many behaviours and avoids under-fitting. The precision is 100% if the model "precisely" allows for the presence of behaviours observed in the log. Figure 5 show that the flower

model lacks precision. This model can replay all the traces in the log, hence fitness is 1. However, it can capture extra behaviours, such as A, B, C, D, E, F, G, H, and S, which have never been seen in the log. As a result, there is a lack of precision.

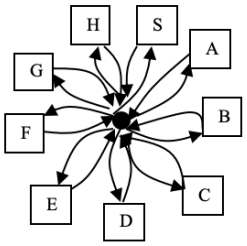


Figure 5. Flower Model

Example 1*:*

Behavioural appropriateness measurement for event log L1 (Table 1) and model M is as follows:

Trace ABDE

Name activity	Num. out edge	Num. used edge
A	2	1
B	2	2
D	2	1
E	1	1
Total	7	5

Trace ACDHF

Name activity	Num. outgoing edge	Num. used edge
A	2	1
C	2	2
D	2	1
H	1	1
F	1	1
Total	8	6

Trace ACHDF

Name activity	Num. outgoing edge	Num. used edge
A	2	1
C	2	2
H	1	1
D	2	1
F	1	1
Total	8	6

Trace ACDGHF

Name activity	Num. outgoing edge	Num. used edge
A	2	1
C	2	2
D	2	1
G	1	1
H	1	1
F	1	1
Total	9	7

Trace ACGDHF

Name activity	Num. outgoing edge	Num. used edge
A	2	1
C	2	2
G	1	1
D	2	1
H	1	1
F	1	1
Total	9	7

$a_B(M1, L1) \approx 1 - X/Y$

where X is,

$[(1207(7-5)/7) + (145*(9-7)/9) + (56*(9-7)/9) + (23*(8-6)/8) + (28*(8-6)/8]$

and Y is,

$(9-1)*(1207+145+56+23+28)$

Hence,

$a_B(M1, L1) \approx 0.966$

Example 2:

Figure 6 displays a flower model whereby all tasks are enabled. The behavioural appropriateness with respect to the above event log is calculated as $a_B(M2, L1) = 0$. This is so because the number of outgoing edge is equal to used edge.

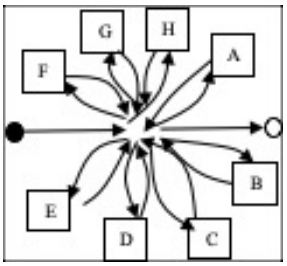


Figure 6. Flower model with lack of precision

C. Simplicity Calculation

Simplicity quantifies the simplicity of the process model. This metric is the only dimension of conformance checking that is not related to the behaviour of the process model or event log. The formula for simplicity used in this study is as follows:

$$\partial s = \frac{|T| - (|T_{DT}| + |T_{IT}|}{T}$$

Assume T is the number of transitions in the model. $T_{DT} \subseteq T$ is the number of alternative duplicate tasks. $T_{IT} \subseteq T$ is the number of redundant invisible tasks. Alternative duplicate consists of tasks that are never repeated together in one sequence. Redundant invisible tasks are tasks that can be removed without changing the behaviour in the model.

Simplicity expresses the complexity of the process model. The simplicity of the model depends on the size of the process model and number of activities in the event log. Simplicity focuses on the size of the process model, and is not related to the behaviour of the process model and event log as well. There are various approaches on how to measure simplicity. In this study, the simplicity metric, which is used, is based on the structural appropriateness related to

control flow of the process model. This metric considers that each activity should be presented at least once during the replay log. It means that duplicate tasks and invisible tasks indicate poor simplicity.

There are two kinds of duplicate task. First, duplicate tasks that have completely different contexts, such as activity A in process models M1, M2, and M3 in Figure 6, which are necessary to specify a certain activity in different contexts. This group of duplicate tasks does not need to be removed from the model and does not affect precision.

Secondly, duplicate tasks that have different contexts of execution in the process model, known as alternative duplicate tasks, which can be tackled in the model, such as activity H in process model M2 that can be executed with "CH" or "GH".

Alternative duplicate tasks are never repeated together in one sequence. There are also two kinds of invisible task. The invisible task in model M1 does not have poor precision because it is created to obtain high fitness, and if it is removed from model M1, then there will be a change in behaviours. The other invisible task is called redundant invisible task. A redundant invisible task can be removed from the model without changing the behaviours, and at that time this group of invisible tasks will have an effect on precision.

There are different structures to express the same behaviour of a process model. For instance, we can consider models M1, M2, and M3 in Figure 7. All three models express the same behaviour of event log L1, but with different structures. Thus, there is a difference in structural appropriateness. Model M1 shows the duplicate task for transition A and an invisible task; both of them have no effect on the precision metric. Model M2 shows the alternative duplicate task of transition "H", duplicate task of transition "A", and a redundant invisible transition. There is a redundant invisible task before transition "B" that can be removed without changing the behaviour. Model M3 shows many duplicate tasks. As a result, the alternative duplicate task and redundant invisible task can reduce structural appropriateness.

Structural appropriateness can be calculated as follows (according to the duplicate task and invisible task):

α is between 0 and 1.

$$T_{DT} + T_{IT} \subseteq T.$$

The duplicate task is always visible. According to the above definition for model M2, there are two alternative duplicate tasks for transition "H" and a redundant invisible task between transitions "B" and "D"; hence, the total transition is 11.

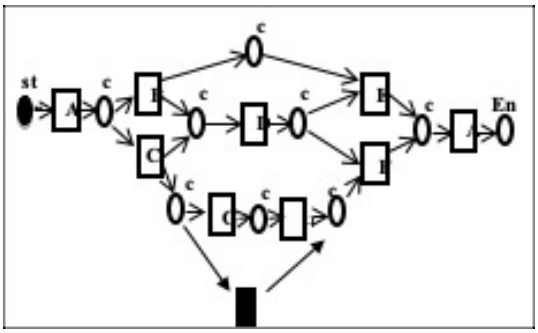
$$\partial s = \frac{11-(2+1)}{11} \approx 0.727.$$

For model M3, all tasks are duplicate except for A, B, and E (A is duplicate but has no alternative duplicate), then

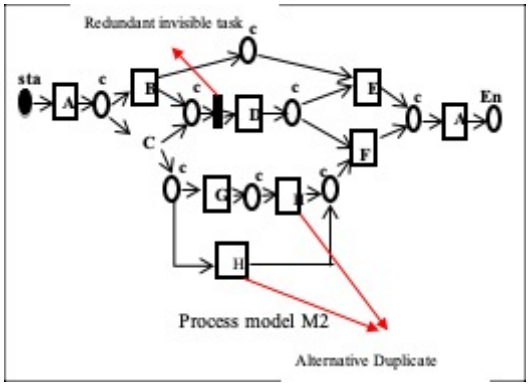
$$\partial s = \frac{31-(19+0)}{31} \approx 0.387.$$

In model M1, if there are not alternative duplicate and redundant tasks, then

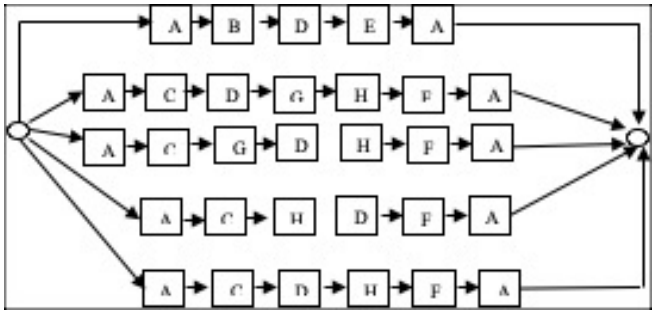
$$\partial s = 1.$$



(a) Process model M1



(b) Process model M2



(c) Process model M3

Figure 7. Three models with the same behaviour

D. Generalization Calculation

Generalisation estimates how well the process model describes the behaviour of the (unknown) system, not only the event log with the observed system behaviour. Generalization can be calculated as follows:

$$Q_g = 1 - \frac{\sum_{nodes} (\sqrt{\#execution})^{-1}}{node\_in\_model}$$

The square root of the number of executions is taken because the effect of having 10 executions instead of 1 is considered a more significant improvement than going from 10 to 100 executions. From each of these values, the power of −1 is taken to normalize it to a value between 0 and 1. Then, these values are summed and divided by the total number of nodes in the tree to gain the average for the whole tree (Buijs, 2014). An example of the generalization metric is described as below based on in Figure 2.

A= 1207+145+56+23+28=1459 →√1459=38.21

$$\partial_G = \frac{1}{38.21} = 0.0261$$

B=1207 → √1207=34.74

$$\partial_G = \frac{1}{34.74} = 0.0287$$

C= 145+56+23+28 → √252=15.8754

$$\partial_G = 0.0630$$

D= 1207+145+56+23+28=1459 →√1459=38.21

$$\partial_G = 0.0261,$$

E= 1207 → √1207=34.74

$$\partial_G = 0.0287$$

G= 145+56 → √201=14.1774

$$\partial_G = 0.0705$$

H= 145+56+23+28 → √252=15.8754

$$\partial_G = 0.0630$$

F=145+56+23+28 → √252=15.8754

$$\partial_G = 0.0630$$

Hence,

$Q_G = 1 - X$, in which X is

0.0261+0.0287+0.0630+0.0261+0.0287+0.0705+0.0630+0.0630

$$= 0.958$$

The four metrics above are computed on a scale from 0 to 1, where 1 is optimal. Replay fitness, simplicity, and precision can reach 1 as the optimal value. All four quality dimension metrics are described in this section. However, there is a question on whether all four quality dimensions are necessary for process discovery. Most of the existing process discovery methods are focused on one or two quality dimension as shown in Table 3, in which a majority of the models are under-fitting or over-fitting.

Table 3. Comparison of the result of process discovery algorithms (f= fitness, p= precision, g= generalisation, s= simplicity).

Algorithm	f	p	g	s
α-algorithm	✗	✓	✗	✗
Genetic miner	✓	✗	✗	✗
Heuristics miner	✗	✓	✗	✗
ILP miner	✓	✗	✓	✗
Inductive miner	✓	✗	✓	✓
Language-based region theory	✗	✗	✗	✗
Multi-phase miner	✓	✗	✗	✗
State-based region theory	✓	✗	✗	✗

For instance, ILP miner has perfect fitness and generalization, but its model is not precise as it produces too many behaviours that are under-fitting. Nevertheless, [19] presented the Evolutionary Tree Miner (ETM) framework that is able to balance these four quality metrics (fitness, precision, simplicity, and generalisation).

Process model M3 in Figure 23 can replay all traces of the event log (high fitness) and it is very precise, but it has poor results for simplicity and generalization. This process model has many duplicate tasks. The process model used in Figure 5 (flower model) has a fitness of 1 and perfect simplicity and generalization results, but it lacks precision. The fitness quality is the most important for process discovery. Fitness estimates the relation between the process model and event log. A process model with a low fitness is unable to provide high precision, generalization, and simplicity. A perfect process model should be able to describe each of the four qualities and can be optimized from all four qualities. The aim of having four perfect quality dimensions is to achieve the best result of these qualities.

4. Conclusion

The aim of this paper is to obtain an overview on process mining and conformance checking. There are still many existing approaches that cannot support some of construct problems. The constructs that cannot be addressed easily by some techniques are: non-free-choice, duplicate task and invisible task. Many approaches assume a one-to-one relation between the tasks in the log and their labels that cannot obtain good results regarding duplicate tasks. After discovering the process model, conformance checking is used to evaluate the process model with the respected event log based on four quality

dimensions of fitness, precision, simplicity, and generalization. Conformance checking techniques need an event log and a model as input. The output consists of diagnostic information showing differences and commonalities between the model and log. All four quality dimensions are defined by formulae and examples in this study. In order to evaluate a process model, all quality dimensions are necessary. As a result, a perfect process model should be able to balance all four quality dimensions of conformance checking. For further works, we plan to investigate the use of learning automata [9],[10],[11] to discover a process model with optimized quality attributes.

Acknowledgements

This research was funded by the Malaysian Ministry of Higher Education with a research grant code of FRGS/1/2016/ICT01/UKM/01/1.

References

[1]. Adriansyah, A., van Dongen, B.F. & van der Aalst, W.M.P. (2011). Conformance Checking using Cost-Based Fitness Analysis. In *IEEE International Enterprise Computing Conference* (EDOC 2011). IEEE Computer Society.

[2]. Adriansyah, A., van Dongen, B. F., & van der Aalst, W. M. (2010, September). Towards robust conformance checking. In *International Conference on Business Process Management* (pp. 122-133). Springer, Berlin, Heidelberg.

[3]. Agrawal, R., Gunopulos, D., & Leymann, F. (1998, March). Mining process models from workflow logs. In *International Conference on Extending Database Technology* (pp. 467-483). Springer, Berlin, Heidelberg.

[4]. De Medeiros, A. A., Van Dongen, B. F., Van der Aalst, W. M., & Weijters, A. J. M. M. (2004). Process mining: Extending the α-algorithm to mine short loops.

[5]. De Medeiros, A. K. A., Guzzo, A., Greco, G., Van Der Aalst, W. M., Weijters, A. J. M. M., Van Dongen, B. F., & Saccà, D. (2007, September). Process mining based on clustering: A quest for precision. In *International Conference on Business Process Management* (pp. 17-29). Springer, Berlin, Heidelberg.

[6]. De Medeiros, A. A., & Weijters, A. J. M. M. (2005). Genetic process mining. In *Applications and Theory of Petri Nets 2005, Volume 3536 of Lecture Notes in Computer Science*.

[7]. de Medeiros, A. K. A., Weijters, A. J., & van der Aalst, W. M. (2007). Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery*, *14*(2), 245-304.

[8]. Cook, J.E. (1996). Process Discovery and Validation Through Event-Data Analysis. *PhD thesis*.

[9]. Esmaeilpour, M., Naderifar, V., & Shukur, Z. (2012). Cellular learning automata for mining customer behaviour in shopping activity. *International Journal of Innovative Computing, Information, & Control*, *8*(4), 2491-2511.

[10]. Esmaeilpour, M., Naderifar, V., & Shukur, Z. (2014). Design pattern mining using distributed learning automata and DNA sequence alignment. *PloS one*, *9*(9), e106313.

[11]. Hadavi, N., Nordin, M. J., & Shojaeipour, A. (2014, June). Lung cancer diagnosis using CT-scan images based on cellular learning automata. In *2014 International Conference on Computer and Information Sciences (ICCOINS)* (pp. 1-5). IEEE.

[12]. Herbst, J. (2000, May). A machine learning approach to workflow management. In *European conference on machine learning* (pp. 183-194). Springer, Berlin, Heidelberg.

[13]. Herbst, J., & Karagiannis, D. (2000). Integrating machine learning and workflow management to support acquisition and adaptation of workflow models. *Intelligent Systems in Accounting, Finance & Management*, *9*(2), 67-92.

[14]. De Leoni, M., & van der Aalst, W. M. (2013, March). Data-aware process mining: discovering decisions in processes using alignments. In *Proceedings of the 28th annual ACM symposium on applied computing* (pp. 1454-1461). ACM.

[15]. Rozinat, A., & Van der Aalst, W. M. (2008). Conformance checking of processes based on monitoring real behavior. *Information Systems*, *33*(1), 64-95.

[16]. Rozinat, A., & van der Aalst, W. M. (2006, September). Decision mining in ProM. In *International Conference on Business Process Management* (pp. 420-425). Springer, Berlin, Heidelberg.

[17]. Pinter, S. S., & Golani, M. (2004). Discovering workflow models from activities' lifespans. *Computers in Industry*, *53*(3), 283-296.

[18]. Petri, C. A. (1962). Kommunikation mit Automaten. Mathematisches Institut der Universität Bonn.

[19]. Buijs, J. C., van Dongen, B. F., & van der Aalst, W. M. (2014). Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *International Journal of Cooperative Information Systems*, *23*(01), 1440001.

[20]. Van der Aalst, W., Weijters, T., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, *16*(9), 1128-1142.

[21]. Weidlich, M., Polyvyanyy, A., Desai, N., & Mendling, J. (2010, June). Process compliance measurement based on behavioural profiles. In *International Conference on Advanced Information Systems Engineering* (pp. 499-514). Springer, Berlin, Heidelberg.

[22]. Weijters, A. J. M. M., van Der Aalst, W. M., & De Medeiros, A. A. (2006). Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, *166*, 1-34.

[23]. Weijters, A. J., & Van der Aalst, W. M. (2003). Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering*, *10*(2), 151-162.

[24]. Wen, L., Wang, J., van der Aalst, W. M., Huang, B., & Sun, J. (2009). A novel approach for process mining based on event types. *Journal of Intelligent Information Systems*, *32*(2), 163-190.

[25]. Wen, L., Wang, J., & Sun, J. (2006, January). Detecting implicit dependencies between tasks from event logs. In *Asia-Pacific Web Conference* (pp. 591-603). Springer, Berlin, Heidelberg.

[26]. Broucke, S. V. (2014). Advances in Process Mining: Artificial Negative Events and Other Techniques.

[27]. Van der Werf, J. M. E., van Dongen, B. F., Hurkens, C. A., & Serebrenik, A. (2008, June). Process discovery using integer linear programming. In *International conference on applications and theory of petri nets* (pp. 368-387). Springer, Berlin, Heidelberg.

[28]. V´azquez-Barreiros, B., Mucientes, M. & Lama, M. (2015). Duplicate Tasks from Discovered Processes, *36th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets*, Brussels, Belgium, June pp. 22-23.

[29]. Sahlabadi, M., Muniyandi, R. C., & Shukur, Z. (2014). Detecting abnormal behavior in social network websites by using a process mining technique. *Journal of Computer Science*, *10*(3), 393-402.

[30]. Sahlabadi, M., Sahlabadi, A. H., Muniyandi, R. C., & Shukur, Z. (2017). Evaluation and Extracting Factual Software Architecture of Distributed System by Process Mining Techniques. *Asia-Pacific Journal of Information Technology and Multimedia*, *6*(2), 77-90.